



Musical Emotions Analysis

TIMOTHY TISMO-CAPILI

CM3203 - One Semester Individual Project – 40
Supervisor: Federico Liberatore

Contents

1	Abstract.....	3
2	Acknowledgments.....	3
3	Introduction	3
3.1	Aims and Objectives.....	4
4	Background	4
4.1	Spotify	4
4.2	Genius	4
4.3	Natural Language Processing	4
4.4	Machine Learning.....	5
4.4.1	Machine Learning Methods	5
4.4.2	Supervised Learning Modelling:.....	5
4.4.3	Multi-Class Classification Methods	6
4.4.4	Evaluation Methods	9
4.5	Tools and Methods	9
4.5.1	Spotify API	9
4.5.2	Genius API	10
4.5.3	NLTK	10
4.5.4	Scikit-learn.....	11
4.5.5	Web Application Tools	11
4.6	Related Works and Studies	12
4.6.1	Klangspektrum	12
4.6.2	Spotify Sentiment Analysis.....	12
4.6.3	Summer hot, Winter not! Seasonal influences on context-based music recommendations.....	13
5	Approach.....	13
5.1	Design Specification	13
5.2	Methodology.....	15
5.2.1	Experimentations with the APIs and NLTK.....	15
5.2.2	Developing an Understanding of Machine Learning	15
5.2.3	Building the Dataset.....	15
5.2.4	Creating, Testing and Evaluating the Machine Learning Models.....	16
5.2.5	Building the Application	16
5.2.6	Deploying the Application.....	16
5.2.7	Testing and Evaluating the Application and Machine Learning Model	16
6	Implementation	17

6.1	Extracting Songs from Spotify Playlists and Building the Dataset	17
6.2	Creating the Models.....	19
6.3	Testing the Models	20
6.4	Building the Web Application	23
6.4.1	Home Page	23
6.4.2	Authentication Page.....	24
6.4.3	Loading Page	26
6.4.4	Results Page	27
6.5	Deploying the Application.....	30
7	Results and Evaluation	31
7.1	Evaluation of the Machine Learning Models	31
7.1.1	Confusion Matrices	31
7.2	User Evaluations.....	33
7.2.1	Evaluation of the SVM Model	33
7.2.2	Evaluation of the Web Application	34
8	Future Work	35
9	Conclusions	36
10	Reflection on Learning	36
11	Appendix	37
12	References	38

1 Abstract

Machine learning is a method of data analysis that grows increasingly important as coding problems grow more complex. One such problem that is quite complex is emotional analysis. The reason why this is a complex task is because the emotions of another person can be hard to determine sometimes even for humans. The task of identifying one's emotions using machine learning is definitely difficult, but not impossible. There are many different ways in which a person can express their emotions and one of them is through music. The intention of this project is to work towards being able to predict the emotions of a person by first being able to predict the emotions that are conveyed in from the songs that a person listens to. The project will study how accurately machine learning can label the emotions that are conveyed by a song by comparing various machine learning techniques. The project will look into getting different people to test the accuracy of the predictions produced by machine learning technique by implementing them into a web application that will be easy for them to access. To achieve these goals, this project will tackle the fundamentals of machine learning, the utilisation of different APIs to retrieve data, and the development of a web application.

2 Acknowledgments

I would like to thank my supervisor Federico Liberatore for providing continuous help and support throughout the project, especially for supporting me when learning machine learning.

3 Introduction

There are many people that experience difficulty when sharing their emotions. Either because they do not want to, or they find that they are unable to share their emotions. Sometimes it is important to express one's feelings because not being able to can be very destructive for the person. People who need to seek mental help may not be able to because they may feel that they are unable to express their feelings explicitly. There is, however, a way that many people do express their emotions without explicitly saying, and that is through the music they listen to.

Many people will agree that listening to music will influence or reflect the mood you are in. Teenagers coming of age would mostly express the way they are feeling through music. Knowing this, the music a person is listening to may be used to discover how a person is feeling. Using machine learning techniques, it may become possible to dissect their emotions from their music.

There are, however, some potential drawbacks and challenges that will have to be faced when trying to get a machine learning model to predict the emotions conveyed from a song. The first one is that there are many songs that convey multiple emotions meaning that it is hard to give it a singular label. The second one is that the emotions conveyed from a song can be subjective. For example, one person could find a song happy and cheerful while the other could argue that there are negative connotations within the deeper meaning of the song, making it actually a sad song.

To tackle the second problem, the solution proposed for this is to use natural language processing. The idea would be to use natural language processing to examine and analyse the deeper meaning of a song. This would be used to help enrich the machine learning model predictions.

3.1 Aims and Objectives

This project's aim is to study and compare machine learning techniques that can help identify the emotions of a person by first finding the emotions that are conveyed through the music they listen to. This work can then be carried on in the future to accurately determine how a person is feeling with their music. In short, the study's main research question is: 'How effective can machine learning identify the feelings conveyed from various songs that a person listens to?'

There are two main goals for this project: the first is to study how effective a machine learning model could predict the mood conveyed from a person's recently played songs, and the second is to develop a fun, interactive app that will give users an insight into their recently played music. The main priority will be the former, however. These two focuses play to each other's strengths since creating an application for a machine learning model will make it easier to get other people to test out the model, and at the same time, using machine learning can be used to enrich the insight that the app will provide over a person's music taste.

The project will require a lot of research into how a mood is represented through music, research on using machine learning techniques and using tools for acquiring data.

4 Background

4.1 Spotify

Spotify is one of, if not, the most popular music streaming service with more than 286 million monthly users as of 19 February 2020. Other popular music streaming platforms like Apple Music (over 60 million monthly subscribers) and Amazon Music (over 55 million monthly subscribers) pale in comparison to Spotify in terms of the number of monthly active users. One of the biggest reasons for this is because of Spotify's music discovery and user recommendation system. Spotify's popularity (especially with the youth) and large database of music is the reason why this specific platform was chosen for this project.

4.2 Genius

Genius is a digital media company that provides access which pro, including songs of different languages. The website allows users to add annotations and interpretations to songs which makes it more unique than other popular song lyric websites such as AZLyrics and Musixmatch. Genius is a well-known company, to the point where it has received a lot of attention from the music industry. It has a YouTube Channel which posts interviews from popular song artists talking about the deeper meanings of the songs.

4.3 Natural Language Processing

Natural language processing (or NLP for short) is section of Artificial Intelligence that deals with the processing and understanding of the human language. The primary goal of NLP is to get programs and computers to understand the human speech and text.

Models will often be used for the analysis of text and speech. One such example of model that is commonly used for processing text is the Bag of Words model. The Bag of Words model counts the occurrences of all the words in a piece of text. How often a word is used and how it is used will be considered and used as a training data for the classifier which then can be used to process new pieces of text.

Tokenization is the process of cutting sentences and text into just words meaning that punctuation and other irrelevant data will be thrown away. The parts that are leftover are converted into tokens which makes it easier for analysing and identifying the data.

Sentiment Analysis

Text written by a person can be perceived as positive, negative, or neutral. Sentiment Analysis is the process of analysing the sentiment of text and determining which of the three it conveys. Sentiment Analysis usually makes use of natural language processing and may also use machine learning or even deep learning algorithms to process the text. Just like with NLP, it uses a model that is trained with previous data to classify new pieces of text.

4.4 Machine Learning

Machine Learning is a subset of Artificial Intelligence where algorithms and programs can improve their capabilities through experience and using data. Machine learning algorithms come mostly in the form of models which can be trained with datasets to classify and predict unprecedented data.

4.4.1 Machine Learning Methods

There are many different methods for implementing models but there are three different categories that machine learning methods fall under: Supervised learning, Unsupervised learning, and Semi-Supervised machine learning.

Supervised learning is where the dataset being used to train the model is already labelled. It relies on previously determined data to classify or predict values or outcomes from new data. Methods that fall under this category include linear regression, logistic regression, random forest classifier and support vector machine.

Unsupervised learning is where the dataset being used to train the model is not labelled. With these types of models, they try to discover hidden patterns or group data together (also known as clustering) to make predictions. This type of learning does not require the need the help of a human and will find patterns or clusters independently.

Semi-Supervised learning is where the data only some data is labelled. This type of learning is a hybrid between supervised and unsupervised learning. It uses a small amount of labelled data and a large amount of unlabelled data during training.

The project will focus primarily on using supervised learning methods.

4.4.2 Supervised Learning Modelling:

There are two main types of supervised methods: Classification and Regression.

Classification is used to make predictions of data that is labelled where the model is to predict a label for new data. Classification problems generally require two or more labels. If a problem involves more than two labels, then it is a multi-class classification problem. When trying to evaluate the score of a classification model performs, it is best done so by calculating its accuracy. This is done by taking the number of correct predictions and dividing it by the number of total predictions. From this, a percentage can be inferred and the higher the number, the more accurate the model.

Regression is used primarily for problems that require continuous, quantitative value predictions. Regression models tend to be made to investigate the relationship between a predictable, dependant variable (such as time) and an independent variable which is the variable that the model is trying to predict.

The project will focus on classifier models; specifically, multi-class classifiers.

4.4.3 Multi-Class Classification Methods

Decision Tree Classifier

A decision tree classifier uses tree data structure, which is made up nodes and branches, to classify data. The structure of a decision tree can be broken down like this:

- The decision nodes are the decisions made in the tree that splits the data into two.
- The branches connect the decisions together and represent the result of each decision.
- The leaf nodes are the end results of the decision tree. This represents the label that the data is to be given and has no branches spanning from it.

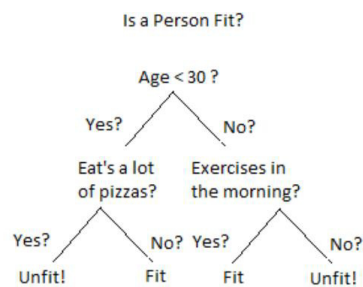


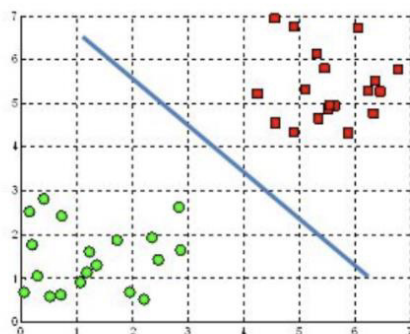
Figure 1 A simplified example of a decision tree classification problem

The example used in Figure 1 describes a classification problem using a decision tree to label if a person is fit or unfit. The questions that are asked are the decision nodes and the labels at the bottom of the structure are the leaf nodes.

Support Vector Machine (SVM)

Support Vector Machines uses an n -dimensional space (where n is the number of features to be used) to plot data using the values from their features as coordinates. It will try to distinguish the different groups (labels) of data by their positioning. It will separate the groups using a hyperplane to split the plane into sections. When trying to make prediction of data items, it will look at the features of the items, plot them on the n -dimensional space and then use the locations of the plot to determine their labels.

A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane

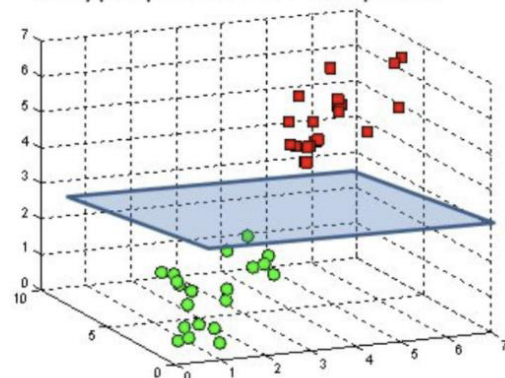


Figure 2 Examples of SVM plots in 2D and 3D space

K-Nearest Neighbours

Similar to SVM, K-Nearest Neighbours plots the training data but this time just on a graph. A datapoint, x , which to be labelled is plotted on the graph. The Euclidean Distance is calculated from x to each of the datapoints on the graph and the k closest datapoints will be selected to determine x 's classification. The label that is most prominent within that selection of datapoints will be assigned to x . To decide what number should be used for k is matter of trial and error. It will vary between different classification problems. The Euclidean Distance formula is as follows:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

\mathbf{p}, \mathbf{q} = two points in Euclidean n -space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n -space

Figure 3 Euclidean distance formula

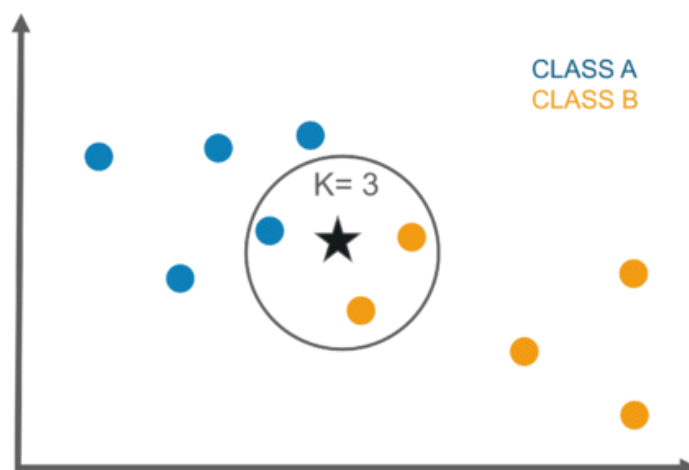


Figure 4 In this example, the star will be assigned to Class B as it is the most prominent label in the circle

Gaussian Naïve Bayes'

The Gaussian Naïve Bayes classifier is largely based on Bayes' theorem in probability theory and statistics. The theorem calculates the probability of an event occurring based off of knowledge of previous events that are similar.

$$P(A | B) = \frac{P(B | A) \cdot P(A)}{P(B)}$$

A, B = events

$P(A|B)$ = probability of A given B is true

$P(B|A)$ = probability of B given A is true

$P(A), P(B)$ = the independent probabilities of A and B

With Naïve Bayes', the events are the classes. It calculates the probability of each class (by dividing the number of values that belong to a class by the quantity of values in the training set) and the conditional probability of a feature value given a class.

To put this in the context of the project, if a song can be labelled as either 'happy' or 'sad' and the audio feature being used to calculate the mood is either 'high energy' or 'low energy', then the conditional probabilities for each label can be calculated as:

$$P(\text{high energy}|\text{happy}) = \frac{\text{number of songs that have high energy AND are happy}}{\text{number of songs that are happy}}$$

$$P(\text{high energy}|\text{sad}) = \frac{\text{number of songs that have high energy AND are sad}}{\text{number of songs that are sad}}$$

$$P(\text{low energy}|\text{happy}) = \frac{\text{number of songs that have low energy AND are happy}}{\text{number of songs that are happy}}$$

$$P(\text{low energy}|\text{sad}) = \frac{\text{number of songs that have low energy AND are sad}}{\text{number of songs that are sad}}$$

The reason why it is called Naïve Bayes' is because it makes the assumption that all the input values are independent of each other. To make predictions using Naïve Bayes', the Bayes' theorem would be like this:

$$MAP(y) = \max (P(x|y) * P(y))$$

Where y is the label to be predicted, x is the set of feature values and the $MAP(y)$ is the likelihood of y. Using this and going back to the example explained earlier, if a song has 'low energy', Naïve Bayes will predict the mood like this:

$$\text{happy} = P(\text{low energy}|\text{happy}) * P(\text{happy})$$

$$\text{sad} = P(\text{low energy}|\text{sad}) * P(\text{sad})$$

Naïve Bayes will calculate the likelihood of each value and the song will be assigned the label that is the highest likelihood.

Gaussian Naïve Bayes is an extension of Naïve Bayes is when the feature values are continuous, and the features are following a Gaussian distribution (normal distribution).

4.4.4 Evaluation Methods

Scoring

The standard method of scoring is to simply count how many labels are accurately predicted and divide that by number of items in a training set.

Confusion Matrix

A confusion matrix shows how many labels of items are being predicted correctly, and how many items are being labelled wrong.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5 Simplified confusion matrix view for a binary classification problem

Leave-One-Out Cross Validation

Since dataset is split into two groups, training data and test data, the results that this brings forth is not representative of the whole dataset because a fraction of the data is being used for testing.

Leave-One-Out Cross Validation (or LOOCV) is an evaluation method that bypasses this issue. The premise of this is to train the model with the entire dataset except for one data point which will be used for to test the model. The result generated by predicting the one data point is ideally stored. This process is repeated for all data points in the dataset. Normal evaluation methods can then be used on all the predictions.

4.5 Tools and Methods

4.5.1 Spotify API

The Spotify API was used to gather create datasets for the machine learning model and to also gather people's Spotify data. To use Spotify's API, a Spotify developer account needed to be created. After an account has been created, an app would need to be created on the dashboard. This would generate the Client ID and the Client Secret which are both used to authenticate requests made to Spotify. This allows the app to then request specific data from Spotify's servers.

If a request involved the data of a user or a group of users, the user would need to login and authenticate the app. For this, a redirect URI would need to be added on the Spotify Developer Dashboard. The redirect URI specifies where the Spotify Authentication service should be redirected to when a user logs in and authenticates the app so that the app can be re-launched.

The Spotify API allows programs to access pretty much any data from their servers as long as it is authorised to do so. The project will primarily focus on retrieving songs from a playlist, retrieving a user's recently played songs, and getting the audio features of a particular song.

Audio features are attributes that are calculated and assigned to each song on Spotify. These can be used for learning and analysing each track. Most of these features are stored as a value between 0.0 and 1.0. These are the definitions of some of the relevant audio features taken directly from the Spotify API documentation:

- *Danceability: Danceability describes how suitable a track is for dancing*
- *Energy: Energy represents a perceptual measure of intensity and activity.*
- *Instrumentalness: Predicts whether a track contains no vocals.*
- *Loudness: the overall loudness of a track in decibels (dB).*
- *Valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.*
- *Tempo: The overall estimated tempo of a track in beats per minute (BPM).*

Using methods that request data from Spotify returns the data in JSON format (JavaScript Object Notation) which is a format that is composed of attribute-value pairs much like a Python dictionary.

Spotipy

Spotipy is a Python library for Spotify's API. This library provides methods and classes that makes requesting Spotify data much simpler and more convenient to access for Python programs that utilises Spotify data.

4.5.2 Genius API

The Genius API is used to retrieve lyrics and annotations of songs for sentiment analysis. To use Genius' API, much like with Spotify's, an account would also need to be created and an app would have to be created on the thereafter. This would then generate a Client ID and a Client Secret which can then be used to make a token to authenticate the app when requesting for song lyrics.

LyricsGenius

Genius API also had a Python library in the form of LyricsGenius which would - just like with Spotipy - provide methods and classes. This makes requesting lyrics more convenient when writing programs that work with using song lyrics.

4.5.3 NLTK

The Natural Language Toolkit (NLTK) is Python library designed specifically to provide methods and classes which helps for building programs that must work with human speech or text.

NLTK can allow for downloading additional packages and models from the internet within the program. One such model that is relevant is the VADER (Valence Aware Dictionary for Sentiment Reasoning) package which also contains. This model was created and trained specifically for the use of sentiment analysis. VADER uses a dictionary that maps words with independent meanings (such

as nouns, verbs, and adjectives) to a sentiment score which represent the intensity of the emotion present.

When parsing a piece of text into the VADER model, it will do the tokenization automatically and will return a compound score which is computed by summing up the valence scores of each word in the parsed-in text, adjusted according to the rules set and then normalised to be between -1 (extreme positive) and +1 (extreme negative). The score returned from this is useful for determining the sentiment of hordes of texts.

4.5.4 Scikit-learn

Scikit-learn is a machine learning library made for Python. It comes pre-loaded with a range of supervised and unsupervised learning algorithms making it great gateway for learning the fundamentals of machine learning.

With the help of the preloaded supervised learning algorithms from Scikit-learn, the steps for creating model are all quite similar:

- Firstly, a labelled dataset needs to be loaded. Datasets are commonly stored as CSV (Comma Separated Value) files.
- Next, the data needs to be pre-processed. What this means is that any irrelevant features should be removed, and the data should be split between data that will be used to train the model and data that will be used to test the model. By default, 75% of the data will be split for training, and the other 25% of the data will be used for testing.
- The selected classifier will be used to fit onto the training dataset.
- Finally, the accuracy of the model is evaluated by having the model used on the test data. The predicted values given by the model are checked against the actual values from the test data.

4.5.5 Web Application Tools

Flask

Flask is a web framework made for Python. It is a simple framework to use for building simple Python. In comparison to other web frameworks like Django, it is considered a micro-framework due to its simplicity and limited functionality. It is mainly used to build the foundation of a web application.

Bootstrap

Bootstrap is a framework designed specifically for the frontend of web applications. It provides an abundance of templates for forms, buttons, tables, navigation, and many other common features that are found on websites today. This was developed to help web developers quickly implement commonly used web features into their site without having to worry too much about CSS or JavaScript. Many companies and application developers resort to using Bootstrap in their websites and web apps.

Heroku

Heroku is a cloud application platform that allows developer to host and deploy their applications and websites to which can then be accessed by anyone the internet. Using this service is free for smaller applications and projects but will be scale for larger projects. An account needed to use this and once one was created, an app would have to be created on the dashboard. After this the flask application can be pushed and deployed from the command line.

Redis

Redis is an in-memory dictionary data store primarily used as a database, message broker, or queue. Redis stores data in the memory rather than a secondary storage such as the hard disk drives or solid-state drives. One of Redis' features is providing workers which can be used to run tasks in the background while the app can focus on other tasks. This is particularly useful for long, on-going tasks.

Heroku provides the option of installing this as an add-on in the form of Redistogo which allows Redis to be used on deployed applications. Just like with Heroku, this option comes free with small applications, but larger ones will have required a monthly payment for.

4.6 Related Works and Studies

4.6.1 Klangspektrum

Klangspektrum is a Spotify application that gives an insight to a user's music by presenting to them the audio features of their top-rated songs. It is presented to them through the use of distribution graphs. This is the application that has brought the inspiration of making an application that gives a person insight into their music taste. However, this application was not built for the intent of emotions analysis as it does not provide any clear insight into how a person is feeling.

4.6.2 Spotify Sentiment Analysis

This post talks about the development of a flask application that would give users an insight into their music the valence of their music. It is mentioned here (from the author's perspective) that the valence value that is retrieved from Spotify can often be inaccurate and does not represent the connotations within certain songs. To enrich the insights provided by the application, sentiment analysis would be applied to the lyrics of a song and the values that are given from this would be visualised using different charts. This post would serve to be a big inspiration for the development of this project. To retrieve the lyrics of a song, the author of the post used Musixmatch API which allowed user to get the lyrics of the song. This, however, limited the capabilities of the application by a little because, as it is mentioned here, Musixmatch only lets users request the first 30% of a song's lyrics in the free version. This means that the results generated from running a sentiment analysis on the song might not be representative of the whole song.

To take the ideas presented in this study further, the application being developed for this project will apply sentiment analysis, not just on the lyrics of a song, but on the annotations of a song as well using Genius. This should further enrich the insight given by the application as it also allows the connotations to be explored deeper by the analysis, providing a more accurate result. Genius also lets users get the entire lyrics for free. The application will not just be limited to showing if songs are positive or negative as machine learning techniques will be used to predict the exact emotion conveyed from a song.

4.6.3 Summer hot, Winter not! Seasonal influences on context-based music recommendations

As this project involves the classification of emotions from music, a lot of research would need to be done to figure out ways that emotions can be classified from music. This particularly study would prove to be very useful as it dives into that specific topic.

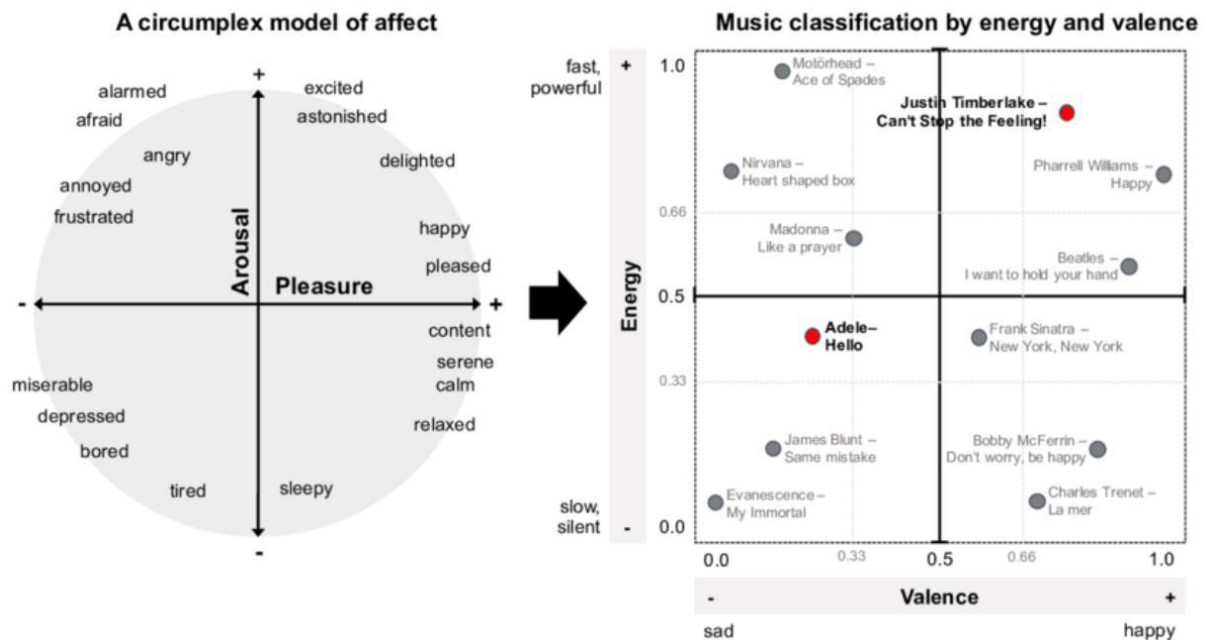


Figure 6 Classification of song by energy and valence

The paper's main focus is to study the correlation between the seasons and the mood presented from the music people listen. It discusses how people's listening habits change depending on the season. In this study, a figure was developed that was derived from the Pleasure Arousal model.

Figure 6 would prove to be useful as point of reference for the classifications in the machine learning model that was developed.

5 Approach

5.1 Design Specification

The application has a simple design overall since the focus of the project is mostly about how accurate the machine learning model is; the main priority was getting the backend of the project to work whilst keeping a clean design. Furthermore, having a simplistic design is considered professional by today's standards of applications and websites.

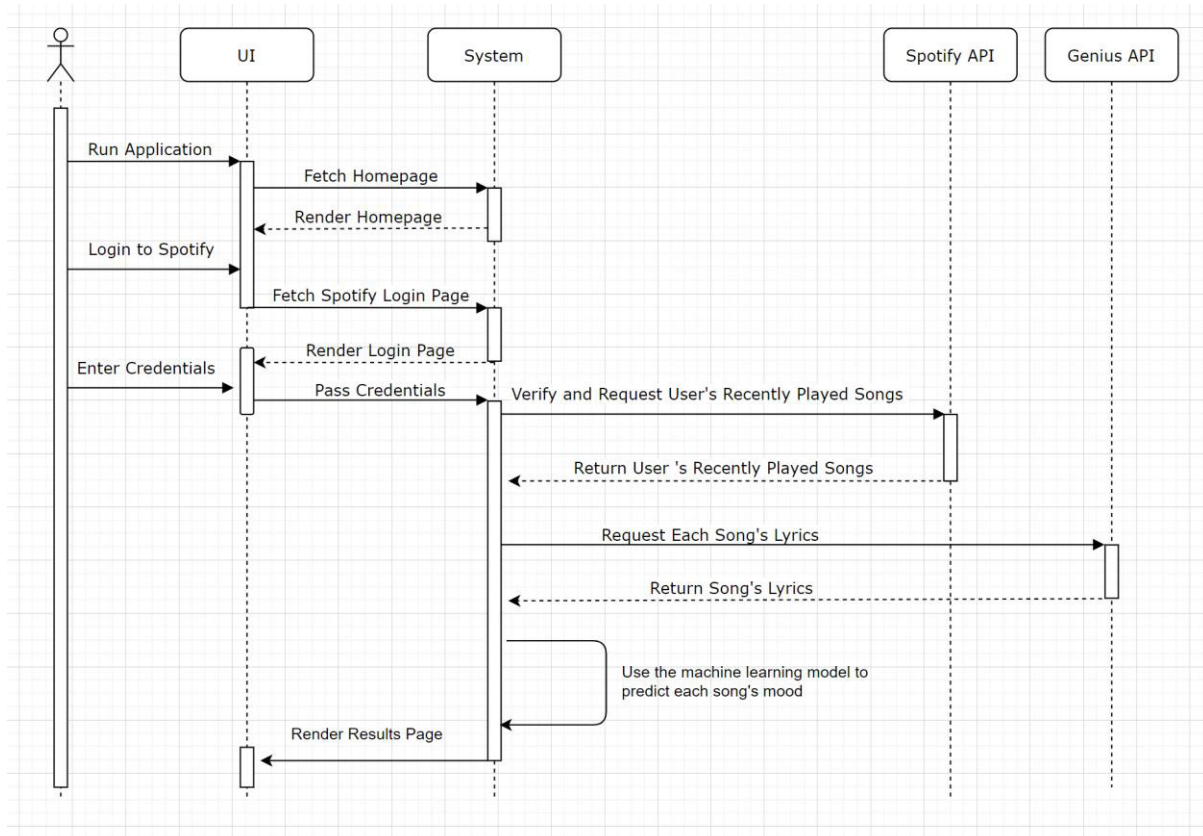


Figure 7: A sequence diagram of the application

Figure 7 gives a visual summary of how the application interacts with the APIs and how the user will navigate from page to the next. Firstly, when a user first enters the application, the user will be greeted by a simple homepage. There will be a button that the user can click on which will bring them to the Spotify login portal where they can enter their credentials and authenticate the application to access their Spotify data.

When the correct credentials are entered and the user gives permission to the application, the application will then send request the user's recently played songs from the Spotify API. Once the songs are retrieved, the application will then request for each song's audio features. At the same time, each song's lyrical and annotation data from Genius API will be requested.

Using VADER sentiment analysis, both the lyrical and annotation data will be analysed by VADER and it will return a value indicating how positive or negative a song is. In conjuncture with each other, the audio features of a song and the values returned by VADER will all be used as features for the machine learning model.

The model will then be called to predict each song's conveyed emotion. Once this is done, the results will be displayed to the user. The app will generate graphs that will provide insights into their emotions such as a chart showing the user how many songs are represented by each mood and/or the valence of the songs overtime.

There will also be a logout button that will be implemented so that other users may log in try the app on the same device.

5.2 Methodology

5.2.1 Experimentations with the APIs and NLTK

The first thing that was needed to be done, was to be familiarised with the Spotify and Genius APIs. Learning Spotify API was first since it the more essential API out of the two. It was required to read the API documentation to learn. Its important to know which data was available to be requested because the potential and the limitations of the application needed to be known at this stage.

Once a good level of understanding was developed for the Spotify API, the Genius API was next. Since Genius was only needed to request for lyrics and annotations, it would not take long to learn.

The next task was to practice using VADER's sentiment analysis. Just like with Genius, VADER sentiment would only serve for one purpose which was to return a value indicating how positive or negative a song's lyrics and annotations are. Therefore, this would also not take long to learn either.

After being proficient enough in both APIs and VADER, the knowledge of the three of them would now need to be combined to create a method that would get a user's recently played songs and retrieve each song's audio features as well as a value from the sentiment analysis lyrics and annotations of the song.

5.2.2 Developing an Understanding of Machine Learning

The next challenge was to be educated with the fundamentals of machine learning. This would require a bit more time to learn as it was not as straight-forward as learning how to work with an API. The Scikit-learn tutorial website was a good place to start as it provides tutorials on learning the basics of machine learning. This would include an introduction of solutions to classification problems. After reading through the website and talking to the project supervisor about how machine learning could be implemented into the project, he would bring attention to multiclass classification and would provide a link to a tutorial on how to implement a number of multiclass classification methods with a dataset. But before this could be done, a labelled dataset would first be needed to use those methods on.

5.2.3 Building the Dataset

There were many Spotify datasets that could be found on websites such as Kaggle, however, none of them were labelled with emotions as this would not be a label provided by Spotify. The solution for this then was to create a completely new dataset that would have each song labelled with a given mood. This would be done by going on Spotify and search for playlists that represented specific emotions.

For example, for songs labelled happy, happy playlists would be searched for. For songs labelled sad, sad playlist would be searched for. This would be done for every emotion that could be easily recognised from a playlist. To make sure that the model predicts accurate and consistent results for each mood, it was important to balance the number of songs that represent each mood. (see Appendix for a full list of playlists that were used).

When a sufficient number of playlists were found for each conceivable emotion, the songs would be extracted from Spotify and it is stored in a file. This would be done using a similar method to the method that was created earlier that would extract a person's recently played songs but instead, the songs of a specific playlist would be extracted. This method would also extract each song's audio features and would also retrieve a sentiment value from each songs lyrical and annotation data. Each playlist would have its own file that it would be stored in so that any mistakes or errors made

during the playlist extraction could easily be amendable in the file. Once all the songs from the chosen playlists were extracted to individual files, the files would all be merged into one file with all the songs from all the playlists. This would form the dataset to be used when making the machine learning model.

5.2.4 Creating, Testing and Evaluating the Machine Learning Models

Scikit-learn's tool will be downloaded and used for the creation of the machine learning models. Since there were multiple multiclass classifier methods that could be used. A model would have to be created for each of the methods. The models will take in, data from the dataset file and use the values as features for the model. Most of the data will be used for fitting the model on to, and the rest of the data will be used for testing and evaluating the accuracy of the model. To evaluate the model, the evaluation methods that were mentioned earlier will all be used.

All of the models will be compared with each other using the results from the evaluation methods mentioned, the model with the best results will be implemented into the application.

5.2.5 Building the Application

After the model has been chosen, an application needed to be built so that the model can be tested by users. The application will be designed as specified earlier. Flask will be used to build the application due to its simplicity to learn and flexibility to work with other packages.

For the front end of both the home page and the results page, bootstrap templates will be adopted into the frontend design since the templates they offer give a professional and simplistic look.

Authenticating the user and generating the results page will primarily be where most of the backend code is. The method that retrieved the user's recently played from Spotify which was developed earlier in the experimentations phase will be implemented here. For each song retrieved, the model will predict the mood of the song using the retrieved song's features, and lyrical and annotation data. The model will compare the new data with the dataset it was fitted with and it will generate a result for each song. Once the model has generated a predicted mood for each song, the website will show the user the mood that the model predicted the most from their music followed by a chart that will show them the quantity of songs that represent each mood. There will also be a table that shows the user's recently played songs along with each song's predicted mood.

5.2.6 Deploying the Application

Heroku will be used to deploy the application since it is free for smaller projects like this and it works with Flask applications as well. An account needed to be created to deploy the application and the Heroku Command Line Interface would also be needed to push the application to their servers.

5.2.7 Testing and Evaluating the Application and Machine Learning Model

Since the app will be tested with other people, the approval of the ethics committee will be needed. This will require having to send in an application form, consent form, participation form and any other files that will be needed for them to approve the testing. Once approval is obtained from the Ethics committee, a broadcast message will be sent out either by email or through social media. This message will let people know that people will be needed to test out the application and will give details about the project and the application. When enough people have contacted and demonstrated their interest, a consent form and participant form will be sent to them to be signed. When they have signed it, they will now be able to test the application out.

To test the application, the participants are asked to look at their recently played songs prior. They will then be asked to pick from three moods from a list of moods; these will be a list of all the labelled moods that are in the dataset. They will then be asked to rank the moods they have chosen from most to least representative. Doing all this will give them an expectation of what mood the application should predict.

Then, they will be instructed to login to the application with their Spotify accounts. In doing so, the application will analyse their music, predict their moods, and show them the results. They are then asked how accurate the model predict their moods and how accurately the model predicts the moods of their music. Finally, they will be asked to give some general feedback on the model and application overall.

The responses from this will be used for analysis and evaluation.

6 Implementation

This section will discuss in detail the implementation of the methodology previously described.

The backend of the application is mostly developed in Python. The application will make use of Spotipy, LyricsGenius, NLTK and Scikit-learn Python packages. Before starting the implementation of the app, a GitHub repository would need to be created at the root of the source folder.

6.1 Extracting Songs from Spotify Playlists and Building the Dataset

Song would need to be extracted from the chosen playlists so that songs a dataset may be formed. This was done in the 'SentimentAnalysisPlaylist.ipynb' file. These were the steps that were taken to retrieve the data necessary to build the dataset.

- Firstly, variables that would authenticate the Spotify application would need to be declared.
 - The username of the account to be accessed is provided.
 - The Client ID and Client Secret are both strings that would be generated from the Spotify developer page.
 - The Redirect URI specifies the URL that the user will be redirected to after logging in and authenticating the app. For now, it is set to redirect the user to a temporary URL. This would be later changed to redirect the user to the results page of the application once it is setup.
 - The scope specifies the data that is to be requested from the user. In this case, the data to be requested is the user's recently played songs.
- These variables would all be used to create the token.
- The Genius access token is generated from the Genius developer dashboard after creating an app there. This would respectively be used to authenticate the application so that it may use the Genius API to request the lyrical data and annotation data of songs.
- The API objects would now need to be created using the tokens that were declared. The objects would be used to call the different methods that would be needed for the extraction of the playlist.
- After everything has been declared, each playlist is retrieved from Spotify using a unique URI code that is assigned to each playlist (see Appendix for a full list of playlists that were used). The number that is assigned to mood value indicates which mood the playlist will be labelled as.
- Extract each song from the playlist and put them into a list.
- For each song:
 - Retrieve the song's audio features

- Search for the song on Genius with the Genius API by passing in the song's name and artist. Retrieve the song's lyrical and annotation data if the song can be found.
- Using VADER, run a sentiment analysis on the lyrical and annotation data of the song.
 - (Sometimes it will fail to run a sentiment analysis on the data. This is either song can not be found on Genius, or the song does not contain lyrics. To solve this issue, a try except statement would be used. If no values were returned from this, a 'null' value will be assigned in the dictionary instead).
- Using the audio features and the values returned by VADER, create a dictionary with all the retrieved values as well as the song name and artists.
- Append the dictionary to a list of dictionaries
- The list will contain all the data needed to be used as features for the model.
- Convert the dictionary into a Pandas data frame. This will be done to make exporting it easier.
- Export the data into a separate CSV file.

```
1047] df = pd.DataFrame(songs)
df
```

	name	artists	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	nlp_lyrics	nlp_annotations	valence+nlp	mood
0	Remember Me	[UMI]	0.45600	0.848	199227	0.344	0.000034	5	0.3500	-8.613	0	0.0374	111.994	4	0.526	0.7286	0.7286	0.5406	Chill
1	North Face	[ODIE]	0.79200	0.882	196800	0.382	0.163000	10	0.0783	-7.356	1	0.0312	99.969	4	0.581	0.9486	0.9486	0.6800	Chill
2	Mine	[Alex Isley, Jack Dine]	0.82800	0.347	212571	0.395	0.000011	6	0.1250	-9.278	0	0.0567	67.492	4	0.133	0.8880	0.8880	0.1908	Chill
3	Shine	[Cleo Sol]	0.68600	0.742	226118	0.504	0.517000	1	0.1030	-10.105	1	0.0392	140.000	4	0.601	0.9515	0.9515	0.6200	Chill
4	Loverboy	[Joeseef]	0.59400	0.356	238123	0.611	0.000000	11	0.1190	-7.219	0	0.0567	79.338	4	0.524	0.9765	0.9765	0.5435	Chill
...
95	Blind Mail	[Kavir Omar]	0.45300	0.890	242196	0.523	0.000002	6	0.0831	-8.526	1	0.0560	111.031	4	0.345	0.9992	0.9992	0.3650	Chill
96	Peaches (feat. Daniel Caesar & Giveon)	[Justin Bieber, Daniel Caesar, Giveon]	0.32100	0.677	198882	0.696	0.000000	0	0.4200	-6.181	1	0.1190	90.830	4	0.464	0.9889	0.9889	0.4838	Chill
97	So Good At Being in Trouble	[Unknown Mortal Orchestra]	0.03630	0.829	230147	0.435	0.878000	0	0.1190	-10.136	1	0.0515	103.816	4	0.594	-0.7326	-0.7326	0.5793	Chill
98	Somehow.	[Phony Ppl]	0.71900	0.582	230973	0.399	0.003630	11	0.6450	-9.934	1	0.0295	92.904	4	0.124	0.2887	0.2887	0.1298	Chill
99	Missing Out	[Syd]	0.00761	0.705	239744	0.528	0.004120	11	0.1290	-5.582	1	0.0416	119.816	4	0.272	-0.9540	-0.9540	0.2529	Chill

100 rows x 19 columns

Figure 8 Example of what the extracted song data looks like in a Pandas data frame

This process will be repeated for all the playlists. When all the playlists have been extracted, the CSV files would then need to be merged to form the dataset.

To merge them, a small program would be made called 'MergeCSV.py'.

```
# Run this to merge all the CSV files in the datasets folder into one
# Make sure to delete any older versions of the files

import os
import glob
import pandas as pd
os.chdir("datasets")

file_pattern = "csv"
list_of_files = [file for file in glob.glob('*.{}'.format(file_pattern))]
print(list_of_files)

#combine all files in the list
combined_csv = pd.concat([pd.read_csv(f) for f in list_of_files ])
#export to csv
combined_csv.to_csv("data.csv", index=False, encoding='utf-8-sig')
```

Figure 9 Python program to merge the CSV files together

6.2 Creating the Models

The models would now need to be created using the Scikit-learn package. A base model would first be created. Since the process for creating a model is the same, this would be used as a template for them to avoid repetition and to make testing the models more convenient.

- The model would first read in the dataset CSV file that was created.
- The data would then need to be pre-processed. This would just eliminate any columns that are not relevant or any columns that cannot be fitted onto the model. Columns have to be integers for it to be fitted on the model. The rows need to be randomised since the rows are ordered by mood in the CSV file. This is done so that the model because the model so that there are no uneven splits when fitting the data.
- The data is then split.
- As this is being done in a Jupyter Notebook, there is a command that is available which is '%store'. This allows variable to be accessed across different Jupyter Notebooks.

```
# Store all variables for different models to use
%store X_train
%store y_train
%store X_test
%store y_test
%store df
```

Figure 10 Variables being stored so that they can be accessed in other notebooks

After the base model is complete, the other models can be made. This would be done in a new Jupyter Notebook for each model.

The first things that needs to be done is to load up the variables that were stored from the base model.

```
#Load variables from BaseModel
%store -r X_train
%store -r y_train
%store -r X_test
%store -r y_test
%store -r df
```

Figure 11 Retrieving the values in a different Jupyter Notebook from the Base Model Notebook

The data would then be fitted onto each respective classifier and will make predictions.

```
# Training a DescisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)
dtree_predictions = dtree_model.predict(X_test)
```

```
# Training a Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(X_train, y_train)
gnb_predictions = gnb.predict(X_test)
```

```
# Training a KNN classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)
knn_predictions = knn.predict(X_test)
```

```
# Training a linear SVM classifier
from sklearn.svm import SVC
svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)
```

6.3 Testing the Models

They are then tested and evaluated against each other. This is done first by calling the standard `score()` method which gives an overall score for how accurately the model predicts each label. The scores of each model would be recorded in a table. When each score has been recorded, the base model code would be ran again so that the it would randomise the rows again and each model can be tested with the same randomised rows. Doing this would make the tests fair as the models are all using the same data.

<u>Aa</u> Name	☰ Test 1 Score	☰ Test 2 Score	☰ Test 3 Score
D-Tree	0.4448938321536906	0.4398382204246714	0.4539939332659252
SVM	0.5803842264914054	0.5824064711830131	0.6056622851365016
KNN	0.43377148634984836	0.42264914054600605	0.4327603640040445
GNB	0.5652173913043478	0.5217391304347826	0.5510616784630941

When using the normal scoring method for getting the accuracy of the model, the method would give scores of about 40-60%. This is because the method only counts the songs that are predicted correctly. Since emotions of music can be subjective, the evaluation needs to be more lenient when checking the accuracy of the model. The solution proposed for this would be to create a custom scoring function.

The scoring function would contain a matrix that would defines the relationship between each mood. Each relationship would be given a value between 0.0 - 1.0; the higher the value, the more similar the mood is. The predicted emotions and the correct emotions would be passed into the function. This matrix was created through a mixture of the Valence-Arousal Figure and using heuristics.

Name	Happy	Sad	Calm	Sleepy	Energised	Aroused	Angry	Chill
Happy	1	0	0.5	0.3	0.5	0.7	0	0.6
Sad	0	1	0.5	0.4	0.3	0.5	0.7	0.5
Calm	0.5	0.5	1	0.8	0	0.5	0	0.9
Sleepy	0.3	0.4	0.8	1	0	0.2	0	0.5
Energised	0.5	0.3	0	0	1	0.3	0.8	0
Aroused	0.7	0.5	0.5	0.2	0.3	1	0.2	0.6
Angry	0	0.7	0	0	0.8	0.2	1	0
Chill	0.6	0.5	0.9	0.5	0	0.6	0	1

Figure 12 Similarity matrix used to define the similarity between the labels

For each predicted emotion and correct emotion, the function would use the matrix to check how similar the emotions are. It would return the value that is assigned to the two emotions and the value would be added to a total sum. The total sum would then be divided by the number of predicted emotions giving a more lenient accuracy score for the models.

As this is a custom scoring method however, the scores produced by this method would need to be validated during the user testing. For the model that is chosen for the web application, the score given by this method for that model would also need to be reflected by the responses that the user gave.

Confusion matrices would also be constructed to better show which labels the model is getting confused with.

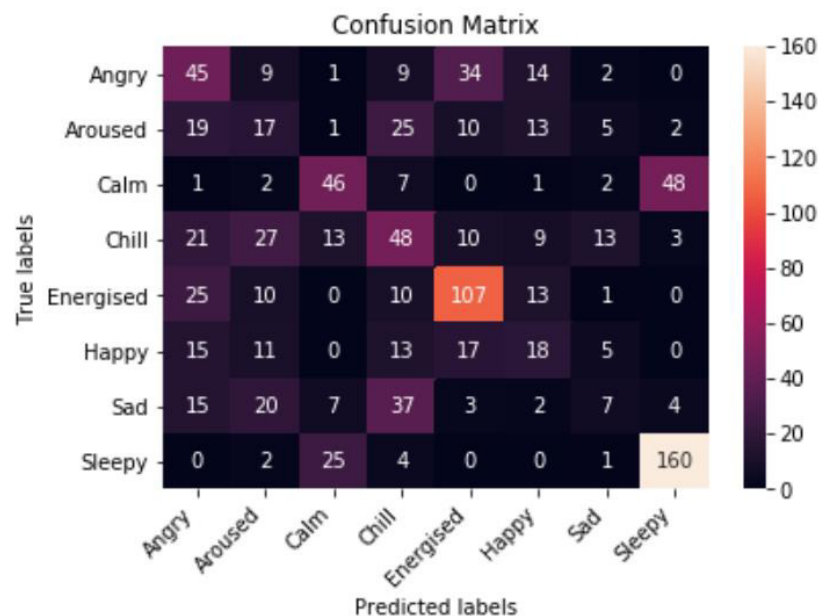


Figure 13 Example confusion matrix showing how many of each labels are being wrongly predicted

```

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Creating a confusion matrix
cm = confusion_matrix(y_test, knn_predictions)

ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax); #annot=True to annotate cells, fmt='g' to disable scientific notation

# Labels, title and ticks
labels = ['Angry', 'Aroused', 'Calm', 'Chill', 'Energised', 'Happy', 'Sad', 'Sleepy']
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(labels, rotation=45, ha='right',); ax.yaxis.set_ticklabels(labels, rotation=0, ha='right');
ax.set_xticklabels

```

Another function that was created for evaluating the models was a function that would count how many moods were labelled correctly for each mood. This would serve to provide a simple view at how many labels are getting accurately predicted.

```

{'Happy': 18,
 'Sad': 7,
 'Calm': 46,
 'Sleepy': 160,
 'Energised': 107,
 'Aroused': 17,
 'Angry': 45,
 'Chill': 48}

```

After the models were tested normally, the same set of evaluation methods would be done again. However, this time it would be tested using LOOCV. When using LOOCV, it would give the same results every time since each row of the dataset is being validated individually, which means that the test does not need to be repeated.

```

for train_ix, test_ix in cv.split(X):
    # split data
    X_train, X_test = X.iloc[train_ix, :], X.iloc[test_ix, :]
    y_train, y_test = y.iloc[train_ix], y.iloc[test_ix]
    # fit model
    model = SVC(kernel = 'linear', C = 1)
    model.fit(X_train, y_train)
    # evaluate model
    yhat = model.predict(X_test)
    print(y_test)
    print(yhat)
    # store
    y_true.append(y_test.iloc[0])
    y_pred.append(yhat[0])

```

The results of the testing would show that the SVM model was the most accurate out of the models at predicting the labels. This would later be elaborated on in the Results and Evaluation section.

6.4 Building the Web Application

To start making the web application, the Flask application file would need to be created first. This would be where most of the backend code would go and where each URL would take the application to.

6.4.1 Home Page

First the home page needed to be created. The home page HTML file would be adopted from a Bootstrap template.

```
<body class="d-flex h-100 text-center text-white bg-dark">
  <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
    <header class="mb-auto">
      <div>
        <h3 class="float-md-start mb-0">VibeCheck</h3>
        {% block nav %}{% endblock %}
      </div>
    </header>

    <main class="px-3">
      <div class="content">
        {% block content %}
        {% endblock %}
      </div>
    </main>

    <footer class="mt-auto text-white-50">
    </footer>
  </div>
</body>
```

Figure 14 Template used for the home page

The Flask framework provided the use of blocks which allows HTML scripts to be placed inside from other HTML files allowing for it to be used as a template file.

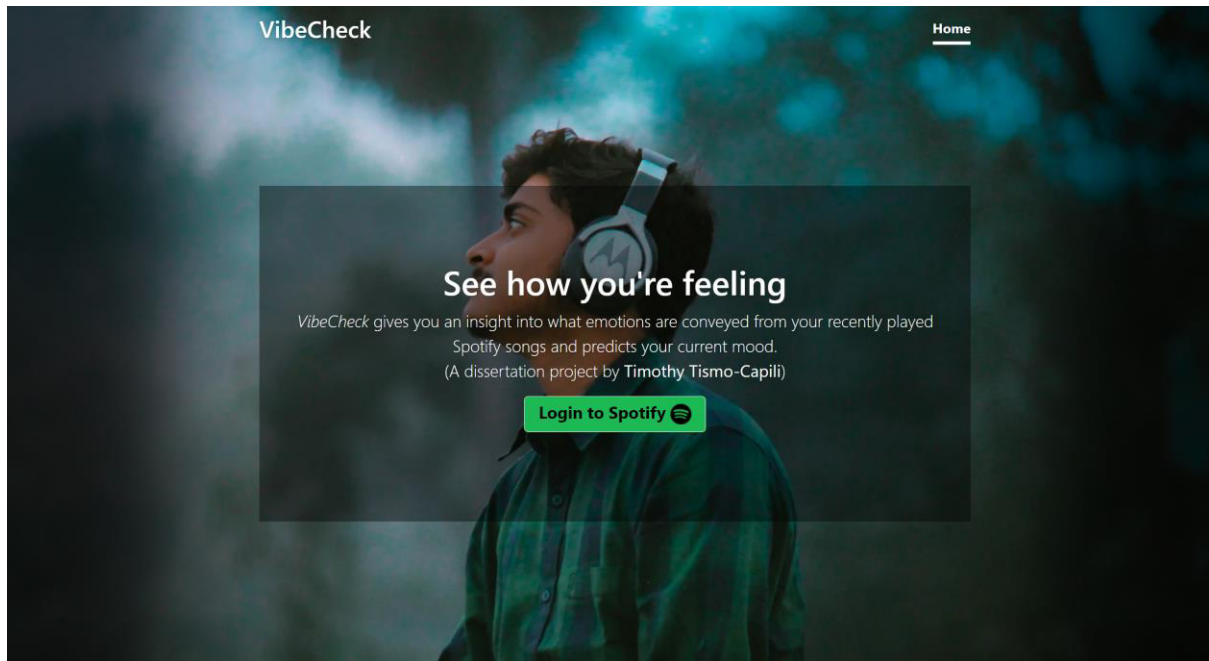
```
{% block content %}
<h1>See how you're feeling</h1>
<p class="lead">
  <em>VibeCheck</em> gives you an insight into what emotions are conveyed from
  your recently played Spotify songs and predicts your current mood.
  <br>(A dissertation project by <b>Timothy Tismo-Capili</b>)
</p>
<p class="lead">
  <a href="login" class="btn btn-lg btn-secondary fw-bold border-white">Login to Spotify
  
  </a>
</p>
{% endblock %}
```

Figure 15 Contents of the homepage

The home page only contains a sentence describing what the application does and then a button that would let the user login to their Spotify. The use of Bootstrap was helpful for keeping the design clean and minimal. Both the home page and results page were made from a Bootstrap HTML template.

The application was given the name 'VibeCheck' which is displayed at the top of the application. The name 'VibeCheck' was chosen because the word 'vibe' is slang for the emotions or atmosphere one is experiencing. It is also a shortened version of the word 'vibrations' which could be directly related to how music produces vibrations, or rather, soundwaves.

A stock image of someone listening to music would be used for the background to make the site more appealing to viewers. There were plans to create a navigation bar for an about page which would more information about the application, and a contacts page where an email and GitHub link would be given. However, due to the time constraints of the project, the focus was put on developing the bare essentials of the website. Since the about page and contacts page were not considered important for the functionality of the website, they would be implemented in the final design of the application.



Once the frontend was finished, the route for the home page needed to be declared in the main Flask application file which would tell the application which template to render when a user is directed to the page.

```
@app.route('/home')
def home():
    return render_template('home.html')
```

6.4.2 Authentication Page

When a user would click on the 'Login to Spotify' button, it would take them to the Spotify authentication portal. From here they can enter their credentials, and the app would show the user which data is going to be accessed which would be necessary for generating the results, to which the user may then choose to authenticate the application.

This was done by redirecting the user to the Spotify authentication page.

```

# get_token(), login(), results(), and api_callback() functions were adopted from this post:
# https://stackoverflow.com/questions/57580411/storing-spotify-token-in-flask-session-using-spotipy
# authorization-code-flow Step 1. Have your application request authorization;
# the user logs in and authorizes access
@app.route("/login")
def login():
    try:
        # Remove the CACHE file (.cache-test) so that a new user can authorize.
        os.remove('.cache')
        session.clear()
    except:
        pass

    # Don't reuse a SpotifyOAuth object because they store token info and
    # you could leak user tokens if you reuse a SpotifyOAuth object
    sp_oauth = SpotifyOAuth(client_id = client_id, client_secret = client_secret,
                           redirect_uri = redirect_uri, scope = scope, show_dialog=show_dialog)
    auth_url = sp_oauth.get_authorize_url()
    print(auth_url)

    return redirect(auth_url)

```

After a user logs in and authenticates the app, the app redirects back to the application with `api_callback()` function which requests an access and refresh token to be used. The access token lets the application request data that it has been authorised to retrieve. The refresh token is used to get a new token when the current one become invalid.

The results would then be generated by the application so that it may be displayed to the user. This is done by first getting the user's recently played songs and the data of the songs which would use the exact same method that was created earlier in the experimentation phase. The model that was the most accurate from the testing is created as well using the same method to create the model. The playlists dataset is loaded in and pre-processed for the model to use. The dataset no longer needs to be split into training data and test data. It would instead use the entire dataset for training the model. The predictions and the user's data are passed back to be used to display to the user.

There is a small issue that would prove to be a problem later with using Genius. Genius takes 2-5 seconds to search and return the data for singular song. The method uses Genius when retrieving the data from the user's recently played songs. Since there are about 50 songs being searched for, this means that the time that Genius takes to search for a song would add up and the method overall would take about 2-5 minutes to complete.

```
def predict(spotifyObject):
    X_train, y_train = getModelValues()

    # Training a linear SVM classifier
    from sklearn.svm import SVC
    svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)

    songs = getSongs(spotifyObject)

    user_data = pd.DataFrame(songs)

    # Drop unnecessary columns
    user = user_data.drop(columns=['duration_ms', 'played_at', 'datetime', 'nlp_lyrics', 'nlp_annotations'])
    X_test = user.iloc[:, 2:15]

    # Predict
    svm_predictions = svm_model_linear.predict(X_test)

    return svm_predictions, user_data
```

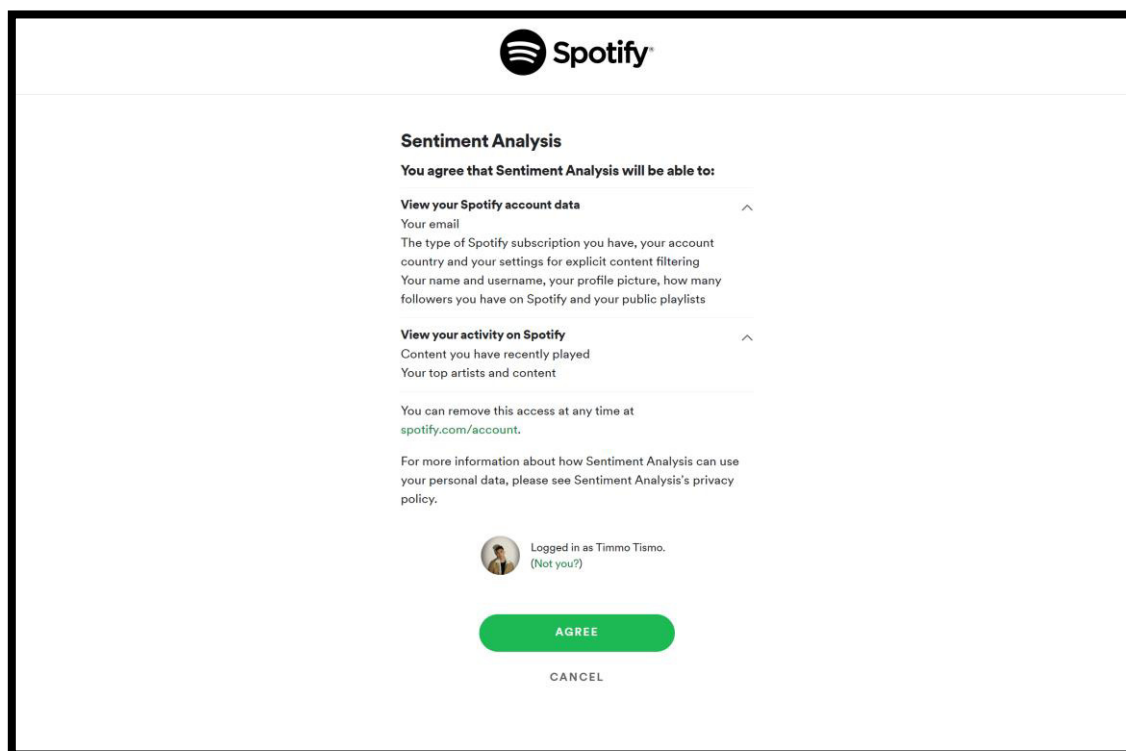


Figure 16 Spotify authentication page

6.4.3 Loading Page

When trying to deploy the application to Heroku, a timeout error kept occurring when trying to generate the results for the user. This is because Heroku sets a time limit of 30 seconds when processing a request and application takes around 2-5 minutes to generate results for a user. Heroku does not allow the timeout value to be adjusted to make sure that deployed applications are responsive and would scale better. A solution that was thought of was to use a Redis background worker which would generate all the results in the background. Then a loading page would need to be made that would periodically refresh and check if the results have been generated.

The process for the loading page is as follows:

- After the user has logged into Spotify and authenticated the app, the user is then redirected to the static loading page that refreshes itself every 5 seconds.
- Whilst that is happening, a Redis worker would generate the predictions in the background
- When the results have been generated, the predictions and the data would be passed in and then processed before passing them into the results HTML template.
- The user is redirected from the loading page to the rendered results template.

6.4.4 Results Page

Just like with the Home page, the front end of the results page would also be adopted from a Bootstrap template. The Bootstrap template provides simple and professional-looking graphs that would be used to display the results to the user. Bootstrap also has some pre-styled tables that would be used to display all the recently played songs to the user along with the mood that was predicted by the model. Bootstrap would also prove to be useful for the overall layout and structure of the application as it allowed it to look nice even if the application were being accessed from a mobile phone. The styling themes used for the results page took inspiration from Spotify's dark theme.

The results page tells the user which mood is most conveyed from their music and would display a pie chart that shows the how many songs were labelled a certain mood. If the user hovers their mouse over the chart, it will show how many songs predicted to convey the specific mood that is being hovered over. There are 2 line graphs that displays the valence and energy of the songs over time which are taken directly from Spotify. Originally, there would only be a single graph with two different coloured lines with one representing valence and the other representing energy. However, when this was implemented, the plots would be quite clustered and was not so easy to read. To make the graphs as coherent as possible, they would be kept separate. The last graph that would be displayed would be a scatter graph showing the relationship between the valence value and energy value of the songs. This took inspiration from the Valence-Arousal model. At the bottom of the page there would be a table showing the user all their recently played songs along with a predicted mood for each song. Finally, there is a logout button that would simply log the user out and redirect them back to the home page.

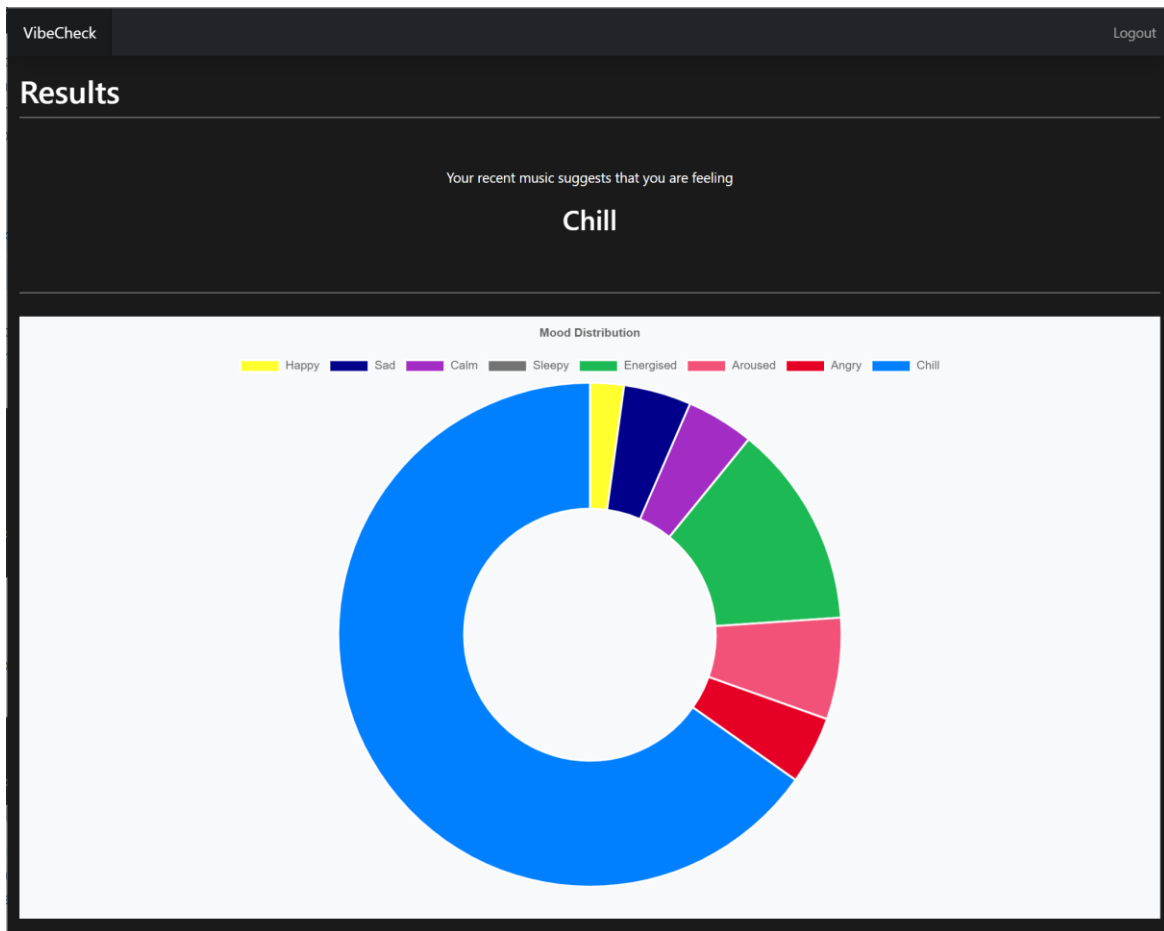


Figure 17 Top of result page showing the result to the user and a mood distribution pie chart

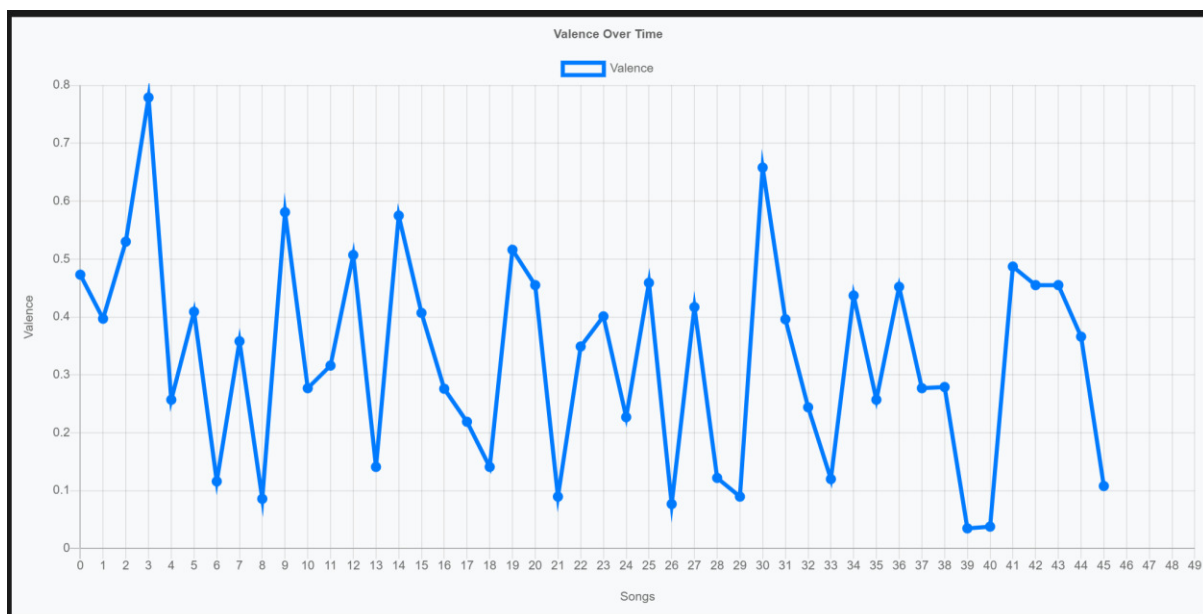


Figure 18 Line graph showing the valence of songs overtime

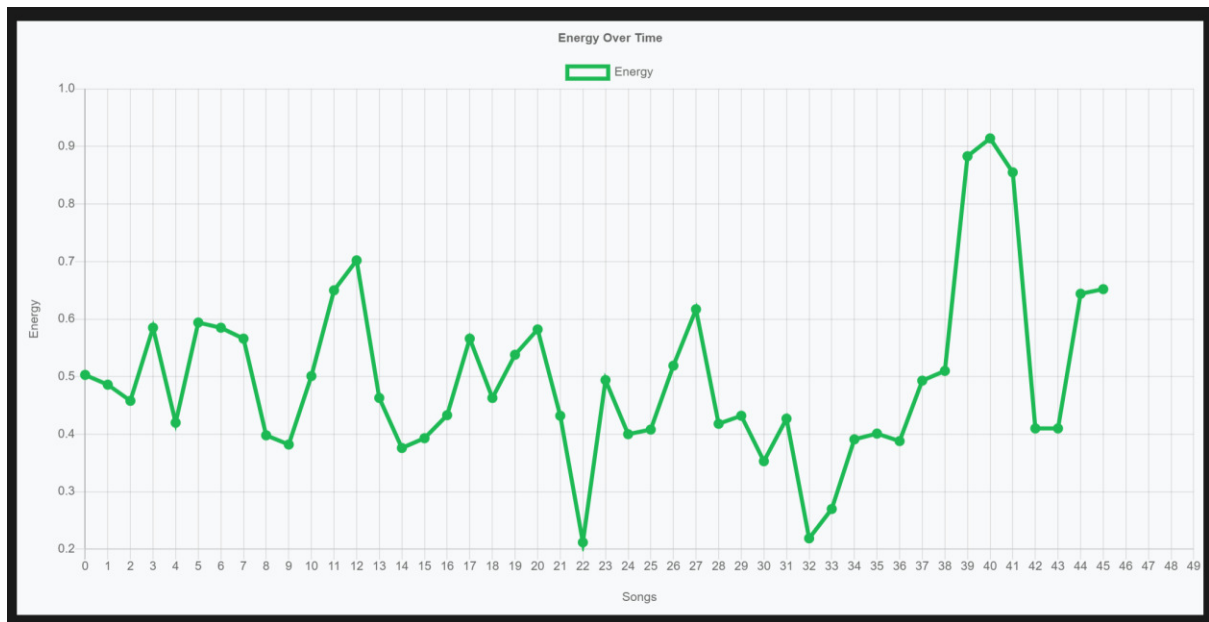


Figure 19 Line graph showing the levels of energy exuded from songs over time

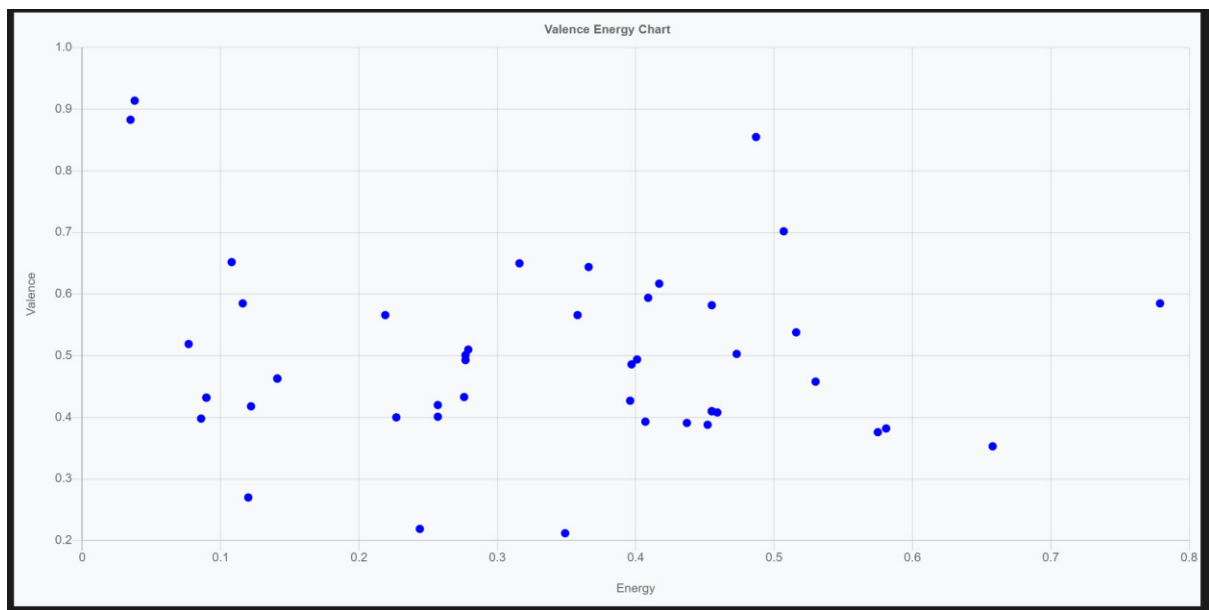


Figure 20 Scatter graph showing the relationship between the valence and energy of the songs

Recently Played					
#	Song	Artist(s)	Time	Date	Predicted Mood
0	Chanel	Frank Ocean	12:10	2021-05-17	Chill
1	Jasmine	DPR LIVE	12:13	2021-05-17	Chill
2	BS	Still Woozy	12:16	2021-05-17	Chill
3	LOVE. FEAT. ZACARI.	Kendrick Lamar, Zacari	12:19	2021-05-17	Happy
4	Brakelights	Omar Apollo	12:22	2021-05-17	Chill
5	Love Galore (feat. Travis Scott)	SZA, Travis Scott	12:27	2021-05-17	Energised
6	Better	Khalid	12:30	2021-05-17	Chill
7	Yeah Yeah	Jaden	12:33	2021-05-17	Chill
8	Myself	NAV	12:37	2021-05-17	Chill
9	North Face	ODIE	12:40	2021-05-17	Chill
10	the reaper	keshi	12:44	2021-05-17	Aroused
11	Sanctuary	Joji	12:47	2021-05-17	Chill
12	Inhale	Bryson Tiller	12:50	2021-05-17	Angry
13	Stuck On You	Giveon	12:53	2021-05-17	Chill
14	Erase	Omar Apollo	12:57	2021-05-17	Chill
15	Dance	offonoff	13:01	2021-05-17	Chill
16	Exchange	Bryson Tiller	13:15	2021-05-17	Chill
17	instagram	DEAN	18:57	2021-05-17	Chill
18	Stuck On You	Giveon	19:00	2021-05-17	Chill
19	SUGAR	BROCKHAMPTON	19:04	2021-05-17	Aroused
20	Walking Home	Mac Ayres	19:07	2021-05-17	Chill

Figure 21 List of recently played songs along with the predicted mood for each song

6.5 Deploying the Application

To deploy the application on to Heroku, the Heroku Command Line Interface needed to be downloaded which allows the files for the application to be pushed from the command line. Gunicorn also needed to be installed which is a Web Server Gateway Interface (WSGI) that is commonly used to run python applications. What needed to be done first was to go to the system's terminal, go to the root directory where all the source code for the project was, and then use the command 'heroku apps: create <insert app name here>'. This will create the application on Heroku. The next step was to create a Procfile which basically tells Heroku what command is needed to start the application. A requirements.txt file is also required as it tells Heroku which Python packages are needed for the application to run. Once this was all setup, the app would then be ready for deployment, which was done by committing and then pushed to Heroku using the following command: 'git push heroku master'. This would then deploy the application to Heroku provided that there were no errors.

The web application can be accessed with this link: <https://vibecheck1.herokuapp.com/home>

7 Results and Evaluation

7.1 Evaluation of the Machine Learning Models

The models that have LOOCV implemented will mostly be discussed here as they result, they show are the most representative of the dataset. The scoring that will mostly be focused on will be the custom scoring method that was created because the score that is generated by this is more representative to the models than the normal scoring method. The score produced by this method will be referred to as the 'Lenient Accuracy' score.

Classifier	Lenient Accuracy	%
Decision Tree	0.7491782553729498	74.91%
SVM	0.8165865992414686	81.65%
K-Nearest Neighbours	0.7173198482932994	71.73%
Gaussian Naïve Bayes'	0.7928445006321114	79.28%

The table of results shows that the SVM model was the most accurate for predicting the emotions beating Gaussian Naïve Bayes' by just 2%. Even though these scores are a lot higher compared to the scores that were calculated from the normal testing methods, the models are still not completely accurate. The models will need a bit more experimenting with in the code, to try to get the accuracy to be higher. For now, the SVM model will be sufficient enough to be used in the web application.

7.1.1 Confusion Matrices

Looking at the decision tree confusion matrix, the model is classifying most of the data as either chill or energised. This shows that the decision tree classifier is not the best for this problem. Ideally, there should be a high concentration of numbers that form diagonal line that goes through top left cell to bottom right cell. This can mostly be seen in the SVM confusion matrix and the Gaussian Naïve Bayes confusion matrix. When examining this line, it is clear to see that the SVM has the most prominent showing line. This proves that the SVM model is the most accurate of the models.

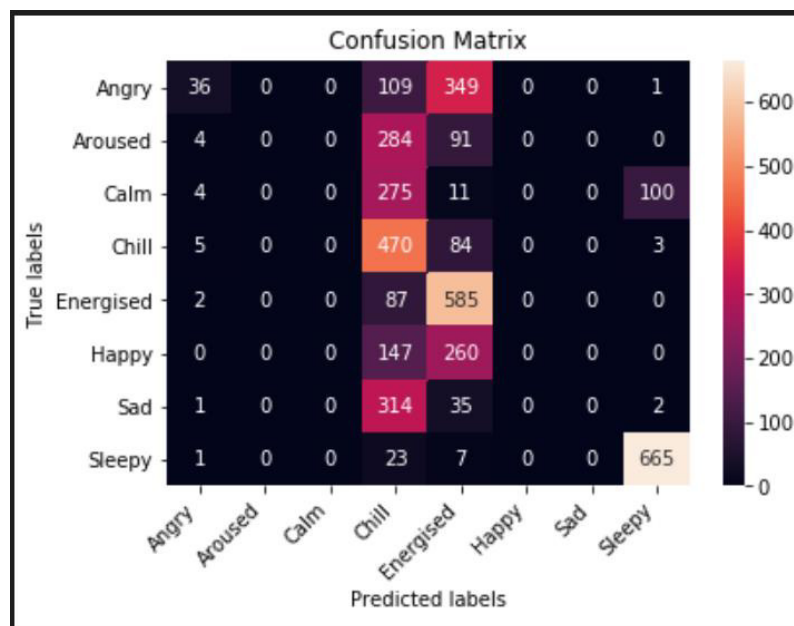


Figure 22 Decision Tree confusion matrix

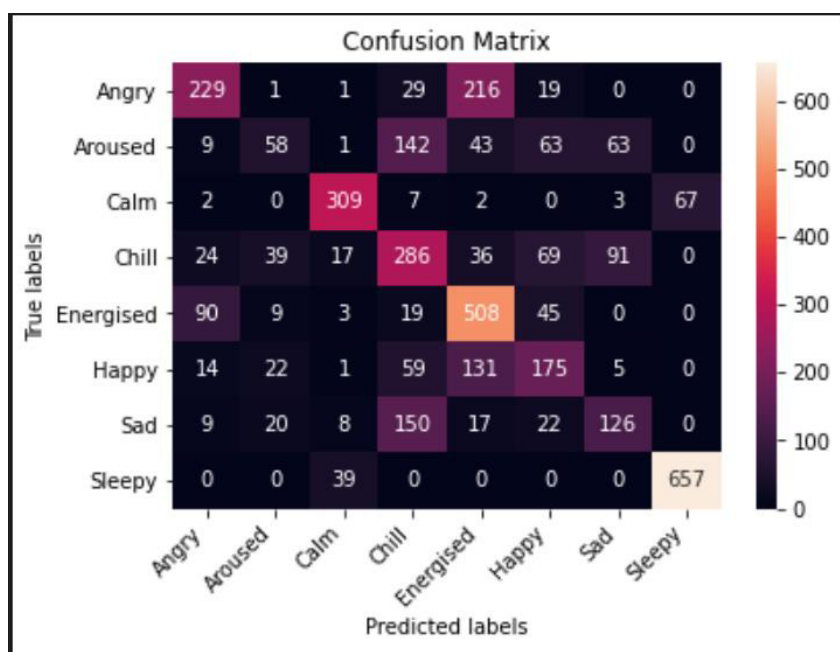


Figure 23 SVM model confusion matrix

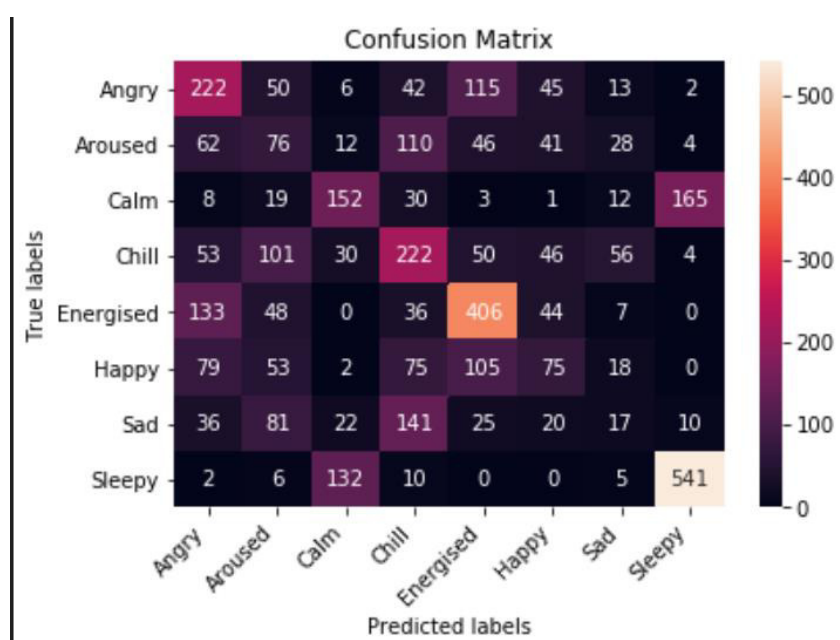


Figure 24 K-Nearest Neighbours confusion matrix

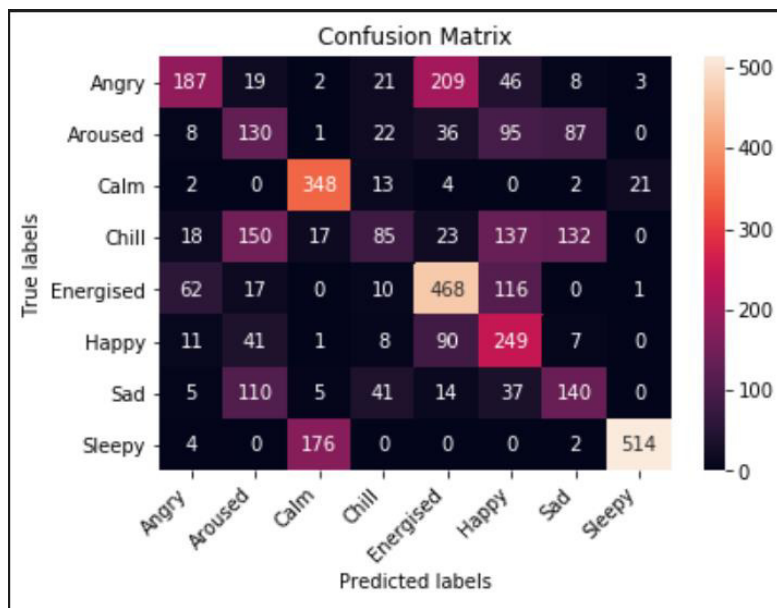


Figure 25 Gaussian Naive Bayes confusion matrix

One thing that is consistent across all the models is that the predictions for 'Sleepy' are always the highest accurately predicted mood. This is probably because the dataset for 'Sleepy' has the greatest number of songs and the music used for it is all mostly very similar. The playlist used to represent most of the emotions would be quite varied in terms of genres. This was done to hopefully get better outcomes for the predictions. However, looking at these graphs, it seemed to confuse the models for a lot of the classifications. Some work might need to be done on the datasets to better improve the results of the models. Energised, chill and calm would all also show to be relatively consistent in some of the models. Happy, sad, and aroused might need ones that are somewhat inconsistent, and their dataset may need some work on.

7.2 User Evaluations

7.2.1 Evaluation of the SVM Model

When asked about the accuracy of the model on a scale of 1-5 (5 being the most), the average response was a 4 out of 5 with some people even saying that the model accurately predicted their mood. This is overall good because this meant that the custom scoring method that was developed for giving a lenient score can be trusted. This also means that the model can be relied on somewhat for predicting the emotions conveyed from different songs.

When reviewing the songs from the table, most of the participants said that they are all mostly accurate with a few exceptions. There were quite a few that mentioned that 'angry' was predicted as the mood for some songs when it should have been labelled as something else. There might be a possibility that the playlists that were labelled as 'angry', may not properly represent the emotion. There was a bit of a struggle trying to find playlists that represented angry because there were not many suitable playlists that were provided by Spotify. When choosing the playlists, it was safest to stick to playlists made by Spotify as they are generally well-maintained and are popular with many users. Spotify, for one reason or another, does not have many playlists that clearly represent 'anger'.

Unfortunately, none of the participants claimed to have 'Sleepy' as one of their representative moods, which means that it was not possible to test validate the accuracy of it being predicted on their songs. On the other hand, all the other emotions had at least one person stating that it was

represented their music which means that they could validate the accuracy of the model when predicting all the other moods.

7.2.2 Evaluation of the Web Application

Overall, the participants seemed really pleased about the functionality and design of the application. Many of them commented on the ‘professional design’ of the app and the fun experience they had with testing it out. Fortunately, no one mentioned any errors when testing the application which meant that the functionality of the app is consistent.

When the participants were asked to give feedback, most of them commented about clarity of the graphs on the results page. The vast majority of them did not understand what the graphs represented and had to be provided an explanation. This was because there was no explanation of what they were presenting which is an oversight on the developer’s side. Besides that, some of them said that the graphs did look ‘cool and professional’

Two of them would also comment on ‘how slow the app can’. They were told prior to testing the application that after logging, it would take 2-5 minute to for the results to load. However, they specified that the home page would take a few minutes to load as well. Upon further inspection into the Heroku console logs, it can be seen that the application goes idle when there is no activity in the application for more than 20 minutes. This is something that cannot really be fixed since that is just Heroku’s way of saving resources.

```
2021-05-24T15:19:16.836016+00:00 app[worker.1]: 15:19:16 Cleaning registries for queue: high
2021-05-24T15:19:16.840471+00:00 app[worker.1]: 15:19:16 Cleaning registries for queue: default
2021-05-24T15:19:16.845225+00:00 app[worker.1]: 15:19:16 Cleaning registries for queue: low
2021-05-24T15:38:19.381765+00:00 heroku[web.1]: Idling
2021-05-24T15:38:19.387303+00:00 heroku[web.1]: State changed from up to down
2021-05-24T15:38:19.431016+00:00 heroku[worker.1]: Idling
2021-05-24T15:38:19.457128+00:00 heroku[worker.1]: State changed from up to down
2021-05-24T15:38:20.799064+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2021-05-24T15:38:20.815346+00:00 heroku[worker.1]: Stopping all processes with SIGTERM
2021-05-24T15:38:20.864716+00:00 app[web.1]: [2021-05-24 15:38:20 +0000] [7] [INFO] Worker exiting (pid: 7)
2021-05-24T15:38:20.865599+00:00 app[web.1]: [2021-05-24 15:38:20 +0000] [8] [INFO] Worker exiting (pid: 8)
2021-05-24T15:38:20.865605+00:00 app[web.1]: [2021-05-24 15:38:20 +0000] [4] [INFO] Handling signal: term
2021-05-24T15:38:20.880892+00:00 app[web.1]: [2021-05-24 15:38:20 +0000] [4] [WARNING] Worker with pid 7 was terminated due to signal 15
2021-05-24T15:38:20.909158+00:00 app[worker.1]: 15:38:20 Warm shut down requested
2021-05-24T15:38:20.934099+00:00 app[worker.1]: 15:38:20 Unsubscribing from channel rq:pubsub:af2a78f376a0417e8f9199a644b2f62c
2021-05-24T15:38:21.118063+00:00 heroku[worker.1]: Process exited with status 0
2021-05-24T15:38:21.177975+00:00 app[web.1]: [2021-05-24 15:38:21 +0000] [4] [INFO] Shutting down: Master
2021-05-24T15:38:21.394680+00:00 heroku[web.1]: Process exited with status 0
2021-05-24T16:07:16.372755+00:00 heroku[web.1]: Unidling
2021-05-24T16:07:16.390858+00:00 heroku[web.1]: State changed from down to starting
2021-05-24T16:07:16.425130+00:00 heroku[worker.1]: Unidling
2021-05-24T16:07:16.426526+00:00 heroku[worker.1]: State changed from down to starting
```

Figure 26 Screenshot of the Heroku console log; shows when the web app becomes idle

# Participant ID	☰ Pre-Mood Ranking	# Accuracy
1	Energised Happy Aroused	3
2	Happy Sad Energised	4
3	Chill Energised Happy	4
4	Happy Energised Chill	4
5	Energised Angry Calm	3
6	Chill Aroused Happy	4
7	Chill Calm Happy	5
8	Calm Chill Happy	4
9	Happy Chill Energised	5
10	Sad Energised Chill	4

Figure 27 Full list of the user test results; moods are chosen and ranked by representativeness, with the mood on the left being the most representative

8 Future Work

An option to give feedback directly to the model i.e., have a button in the table of songs labelled something like ‘this song does not convey this emotion’ and clicking that button will make a pop-up form appear where the user can then select the proper emotion that they feel the song conveys. This could provide the model with the chance to grow and become more accurate.

The addition of more emotions as well could benefit people more as well since there are more than 8 emotions that can be conveyed through music. This would require searching for more playlists to use as a dataset.

Other quality of life features such as being able to see which song is being represented on the graph when hovered over. It is possible to determine the songs represented on the graph by hovering over a plot point, looking at the number and then finding the song that correlated to the number on the list. However, this is a very tedious process, and the user would much benefit from just being able to see the song on the graph without it being visually cluttered.

Different approaches to the problem could be explored next time, such as using a multi-label classification model or maybe even going for a unsupervised or semi-supervised approach.

The code could benefit more from being optimised. One such method for this would be to maybe use a different API for retrieving lyrics. Using the Genius API made the application slower because it had to search for songs individually. Although it was slightly faster when deployed onto Heroku, it would be nice if the user only had to wait a maximum of 10 seconds for the results page to load. Doing so might encourage them to try it out more often. There were considerations of also using the

user's top-rated tracks or most recently liked songs. However, adding more songs for the application to request from will make loading times longer. Currently, there are only 50 songs that are being requested which only takes a 2-5 minutes for results to be generated. Any longer would potentially discourage users from using the application.

9 Conclusions

The application was able to mostly please the participants upon testing and there did not seem to be many issues outside of the graphs not having much explanation. Other than that, there were no errors reported and the data seemed to be presented well to the participants.

The graphs that were implemented did not seem to provide the quality of insight to the user that was initially intended. If the graphs had an explanation, the user would probably get more enjoyment from using the application.

The machine learning model seem to also be mostly successful in predicting the emotions conveyed from a person's music. There definitely needs to be improvements made to the model and the dataset, but for a first attempt, this a great start.

Overall, the project was a success. Both of the main aims of the project were achieved to a sufficient manor. This whole study has proven that it is possible to use machine learning techniques to predict the emotions of music conveyed through their music.

In future, the results could potentially be expanded and improved upon in future works to directly predict the mood of a person.

10 Reflection on Learning

This project was challenging to say the least. Requiring me to learn how to do several things like working with an API, working with natural language processing, understanding, and implementing machine learning techniques, developing both frontend and backend developing skills, and learning how to deploy the application. There was a lot of research that needed to be done for this project and it constantly challenging me every step of the way. However, I like the challenge that this project provided as it pushed me to continuously think of solutions and workaround for different problems.

Overall, I am quite proud of what I have achieved with this project as it combined different aspects of computer science into one project.

11 Appendix

Full list of playlists used for the dataset.

Playlist Name	Mood	No. of Songs
Happy Hits!	Happy	100
Good Vibes	Happy	133
Mood Booster	Happy	74
Feelin' Good	Happy	100
Sad Songs	Sad	60
All The Feels	Sad	100
Life Sucks	Sad	100
Idk.	Sad	100
Atmospheric Calm	Calm	93
Calm	Calm	100
Deep Focus	Calm	196
Sleep	Sleepy	186
Deep Sleep	Sleepy	201
Night Rain	Sleepy	317
Energy Booster: Dance	Energised	100
Beast Mode	Energised	200
Beast Mode Rock	Energised	100
Motivation Mix	Energised	100
Beast Mode Hip-Hop	Energised	76
Drum and Bass Top 100	Energised	100
Sexy R&B	Aroused	55
Sexy as Folk	Aroused	49
Love Pop	Aroused	100
Timeless Love	Aroused	100
Bedroom Jams	Aroused	74
Rage Beats	Angry	100
Complete Chaos	Angry	100
New Metal Tracks	Angry	100
Angry rap	Angry	188
Chill Hits	Chill	207

Chill Vibes	Chill	50
Chill R&B	Chill	75
Indie Chillout	Chill	100
Lowkey	Chill	

12 References

Scikit-learn.org. 2021. Supervised learning: predicting an output variable from high-dimensional observations — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html> [Accessed 28 May 2021].

Musically.com. 2021. How many users do Spotify, Apple Music and streaming services have?. [online] Available at: <<https://musically.com/2020/02/19/spotify-apple-how-many-users-big-music-streaming-services>> [Accessed 28 May 2021].

Monkeylearn.com. 2021. [online] Available at: <<https://monkeylearn.com/sentiment-analysis/>> [Accessed 28 May 2021].

Medium. 2021. Your Guide to Natural Language Processing (NLP). [online] Available at: <<https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1>> [Accessed 28 May 2021].

Brownlee, J., 2021. Difference Between Classification and Regression in Machine Learning. [online] Machine Learning Mastery. Available at: <<https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>> [Accessed 28 May 2021].

Brownlee, J., 2021. Naive Bayes for Machine Learning. [online] Machine Learning Mastery. Available at: <<https://machinelearningmastery.com/naive-bayes-for-machine-learning/>> [Accessed 28 May 2021].

code), U., 2021. SVM| Support Vector Machine Algorithm in Machine Learning. [online] Analytics Vidhya. Available at: <<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>> [Accessed 28 May 2021].

Education, I., 2021. What is Machine Learning?. [online] Ibm.com. Available at: <<https://www.ibm.com/cloud/learn/machine-learning>> [Accessed 28 May 2021].

Edureka. 2021. KNN Algorithm using Python | K Nearest Neighbors Algorithm | Edureka. [online] Available at: <<https://www.edureka.co/blog/k-nearest-neighbors-algorithm/>> [Accessed 28 May 2021].

En.wikipedia.org. 2021. Euclidean distance - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Euclidean_distance> [Accessed 28 May 2021].

En.wikipedia.org. 2021. Genius (website) - Wikipedia. [online] Available at: <[https://en.wikipedia.org/wiki/Genius_\(website\)](https://en.wikipedia.org/wiki/Genius_(website))> [Accessed 28 May 2021].

GeeksforGeeks. 2021. Multiclass classification using scikit-learn - GeeksforGeeks. [online] Available at: <<https://www.geeksforgeeks.org/multiclass-classification-using-scikit-learn/>> [Accessed 28 May 2021].

know!, 7., 2021. Regression Techniques in Machine Learning. [online] Analytics Vidhya. Available at: <<https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/>> [Accessed 28 May 2021].

Medium. 2021. Decision Tree Classification. [online] Available at: <<https://medium.com/swlh/decision-tree-classification-de64fc4d5aac>> [Accessed 28 May 2021].


Medium. 2021. Support Vector Machine — Introduction to Machine Learning Algorithms. [online] Available at: <<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>> [Accessed 28 May 2021].

12.1 References

Brownlee, J., 2021. A Gentle Introduction to Scikit-Learn. [online] Machine Learning Mastery. Available at: <<https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/>> [Accessed 28 May 2021].

Brownlee, J., 2021. A Gentle Introduction to Scikit-Learn. [online] Machine Learning Mastery. Available at: <<https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/>> [Accessed 28 May 2021].

DigitalOcean. 2021. How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK) | DigitalOcean. [online] Available at: <<https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk>> [Accessed 28 May 2021].

- GitHub. 2021. cjhutto/vaderSentiment. [online] Available at:
<<https://github.com/cjhutto/vaderSentiment>> [Accessed 28 May 2021].
- Klangspektrum.de. 2021. Home. [online] Available at:
<<https://www.klangspektrum.de/>> [Accessed 28 May 2021].
- Medium. 2021.  Spotify Sentiment Analysis. [online] Available at:
<<https://towardsdatascience.com/spotify-sentiment-analysis-8d48b0a492f2>>
[Accessed 28 May 2021].
- Ncbi.nlm.nih.gov. 2021. Home - PMC - NCBI. [online] Available at:
<<https://www.ncbi.nlm.nih.gov/pmc/>> [Accessed 28 May 2021].
- Raw.githubusercontent.com. 2021. [online] Available at:
<https://raw.githubusercontent.com/axsauze/reddit-classification-exploration/master/data/reddit_train.csv> [Accessed 28 May 2021].
- Techopedia.com. 2021. What is the Natural Language Toolkit (NLTK)? - Definition from Techopedia. [online] Available at:
<<https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk>>
[Accessed 28 May 2021].